COOPERATIVE NETWORKING

Edited by Prof. Mohammad S. Obaidat and Dr. Sudip Misra

Book Chapter

COOPERATIVE INTER-NODE AND INTER-LAYER OPTIMIZATION OF NETWORK PROTOCOLS

Authors:

Dzmitry Kliazovich

DISI - University of Trento
Via Sommarive 14, I-38050
Trento, Italy
Email: kliazovich@disi.unitn.it

Fabrizio Granelli

DISI - University of Trento
Via Sommarive 14, I-38050
Trento, Italy
Email: granelli@disi.unitn.it

Nelson L. S. da Fonseca

State University of Campinas
SP, Brazil
E-mail: nfonseca@ic.unicamp.br

Abstract

This chapter introduces a novel concept of cooperation among network nodes based on inter-layer and inter-node relationships with the aim of dynamically tuning the TCP/IP protocol parameters for performance enhancement. In such interaction, cooperative decisions are made to find optimal setup of the values of the parameters of these protocols by considering their past operation experience.

Results evince that cooperation between layers of a protocol stack can bring significant improvements in data transfer performance when compared with non-cooperative approaches. One of the main points behind the design of the proposed cooperative optimization framework is the ability to tune configuration parameters in runtime and in a distributed fashion.

I. INTRODUCTION

End-to-end Quality of Service (QoS) provisioning and network performance optimization under changing network environments imply on challenges which solutions demanding coordination of protocols and nodes and joint optimization of the protocol parameters. The need for such coordination is a direct consequence of the designs of the TCP/IP and ISO/OSI architectures which relay on the separation of scope and objectives of the various layers. Such design philosophy implies on the absence of coordination among the protocols operating at different layers.

Moreover, the widespread diffusion of the TCP/IP reference model has only strengthened interaction among protocols allowing only evolutionary rather than a revolutionary approach.

In this scenario, dynamic adjustment and optimization of the parameters of the protocol stack through inter-node and inter-layer cooperation represents a feasible option to support fine-tuning of parameters and enhancement of network performance.

The introduction of cooperation among network nodes to enable flexible adaptation of operating parameters was envisaged by J. Mitola III [1] with the introduction of the concept of *cognitive radio* – aimed at providing efficient spectrum sharing by cooperative and adaptive access to the available transmission resources. Actually, cognitive radio is concerned with the tuning of parameters at physical and link layers as well as optimization goals at these layers. The broader concept of *cognitive network* was introduced to cope with system-wide goals and cross-layer design and can be considered a generalization of the cognitive radio concept. A cognitive network involves cognitive algorithms, cooperative networking, and cross-layer design in order to provide dynamic configuration and real-time optimization of communication systems.

The main contribution of the chapter is an analysis of cooperative inter-node and inter-layer networking issues and solutions from an architectural point of view. Moreover, a framework for cooperative configuration and optimization of communication protocols performance is introduced.

The proposed architecture is concerned not only with the initial setup of protocol parameters, but also with their timing, reconfiguration, and optimization during the network runtime. The architecture requires the introduction of a cognitive plane

operating “in parallel” with the protocol layers and which is capable of monitoring each protocol layer parameter as well as controlling them by issuing configuration commands. For the duration of the optimization process, the cognitive plane monitors the feedback from all the protocol layers, which includes reports on the values of target parameters. For example, the metric at the physical layer can be the obtained data rate, while at the application layer the feedback metric can be the perceived quality of real-time multimedia applications.

II. A Framework for Cooperative Configuration and Optimization

2.1 Tuning TCP/IP Parameters

The TCP/IP reference model [2] is the “de facto” standard for communication in the Internet. It contains a large variety of protocols, whose parameters need to be adequately set for proper functioning. Table I presents a snapshot of the most widely used protocols, their main configuration parameters and corresponding performance metrics the parameters affect.

TABLE I. Main TCP/IP Protocol Stack Configuration Parameters

Protocol layer	Protocol name	Parameter	Metric
Application	File Transfer (FTP)	Number of parallel transfers	FTP Goodput
	VoIP	• Coding rate • Coding interval • FEC strength	• Mean Opinion Score (MOS)
	Video Streaming	• Streaming bitrate • Frame Rate • Keyframe interval	• Peak signal-to-noise ratio (PSNR) • Structural Similarity (SSIM) • Video Quality Measurement (VQM)
Transport	Transmission Control Protocol (TCP)	• Congestion window (w) • Slow start threshold (ssthreshold) • Aggressiveness of window increase (α) and decrease (β) • Protocol version	• Flow Goodput
Network	Routing	• Routing type	• Route setup delay • End-to-end delay
Link	MAC	• Contention window (cwnd) • Fragmentation threshold (MTU) • Retransmission scheme	• Data rate provided to higher layers • Medium access delay
Physical	PHY	• Transmit rate, power • Type of modulation, coding • Frequency channel	• Transmission rate • Bit error rate

Application layer provides the environment for supporting and running user applications. Configurable parameters and quality metrics at this layer depend on the nature of applications. For File Transfer (FTP) applications [3], a configurable parameter could be the number of parallel connections and the main quality metric is the file transfer goodput. For Voice over IP (VoIP) applications controlling coding rate, coding interval, and Forward Error Correction (FEC) [4] impact the voice quality commonly expressed using Mean Opinion Score (MOS) metric [5].

For video streaming applications, streaming bitrate, framerate, and keyframe interval determine the quality of video flow perception. High bitrate values allow video transmissions with high resolutions, high framerate values improve perception of the video samples involving high motions, while shorter keyframe intervals improve decoding capabilities in the presence of frame losses or transmission errors.

Transport layer is generally represented by Transmission Control Protocol (TCP) [6] and User Datagram Protocol (UDP) [7] protocols. While UDP is lightweight, with the main task of providing differentiation of IP datagrams between different port numbers, TCP implements complex mechanisms to achieve reliable data transfer, including ARQ, flow control and congestion control.

Indeed, congestion window of TCP connection is the main mechanism controlling outgoing rate and is a key in window evolution strategy. Controlling the aggressiveness of congestion window increase factor (α) and decrease factor (β) allows adjusting the tradeoff between network utilization, protocol fairness, and the level of network congestion improving data goodput performance.

Network layer is responsible for routing packets across interconnected networks or network domains. Hence the main performance metric is the quality of the selected route expressed as the number of hops or end-to-end delay.

Link layer serves network layer requests and controls the access to the actual transmission medium. Most of the controllable parameters in this layer are determined by the communication technology in use. In Career Sense Multiple Access (CSMA) protocols, such as WiFi IEEE 802.11 [8] or Ethernet IEEE 802.3 [9], the main tunable parameters correspond to the size and evolution of the contention window, as well as the retry limit, which corresponds to the maximum number of retransmission attempts taken at the link layer before a packet is discarded.

Physical layer parameters are defined by the nature of the transmission medium. Wireless access technologies can provide control over the power level of the transmitted signal, choice of the type of modulation, and frequency channel allocation. Physical layer performance can be defined in terms of the data rate achievable, as well as Bit Error Rate (BER) achieved for the transmitted bit stream.

As one can noticed, several parameters exist in the TCP/IP stack that can be tuned, and, even more important, some of them can impact overall applications performance. For instance, the parameters governing the TCP transmission window and the initial value of the window have great influence on the performance, especially in networks with high bandwidth delay product [11].

2.2 Cooperative Optimization Architecture

Fig. 1 outlines the main functional blocks of an architecture enabling cooperative optimization of the values of protocol parameters. To some extent, the proposed architecture is a potential instantiation of the concept of cognitive network under the constraint of enforcing given by interoperability with existing TCP/IP stack.

To operate with a standard protocol stack, each protocol layer is enhanced with a small software module which should be able to both obtain information internal to the specific layer (observation) or to tune its internal parameters (action). The information sensed at different protocol layers is delivered to the cognitive plane implemented at the cognitive node. The cognitive plane performs data analysis and decision making processes.

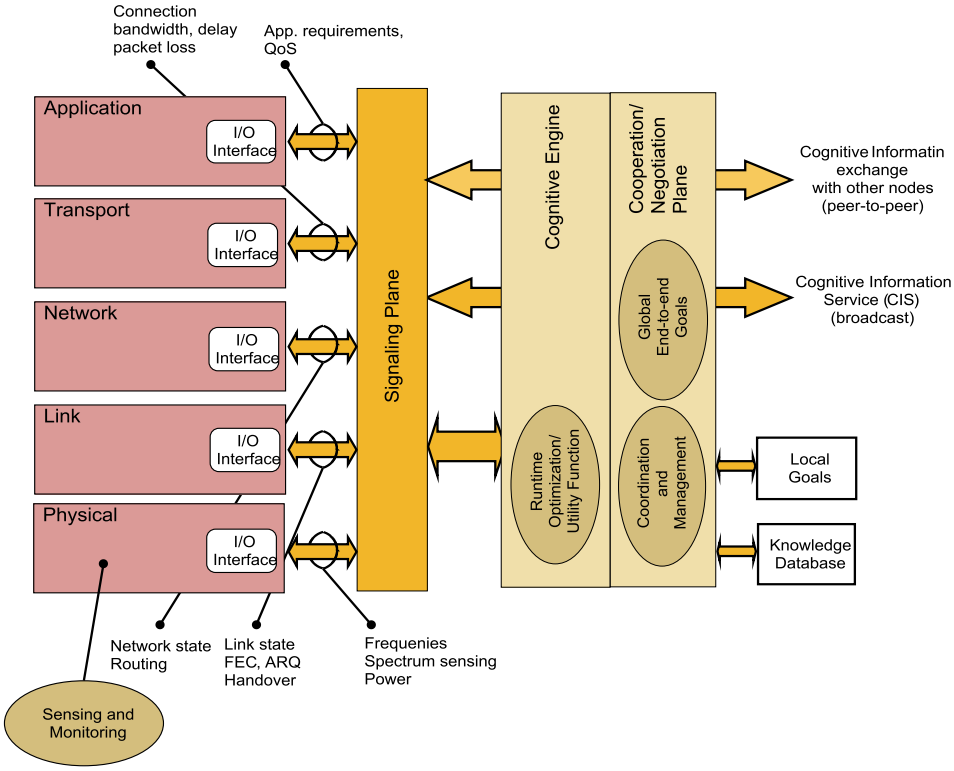

Fig. 1. Cooperative Framework.

Results of data analysis could lead to information classified as knowledge which is storable in the local knowledge database.

The main task of the *cognitive engine* at every node is the optimization of different protocol stack parameters in order to converge to an optimal operational point given the network condition. The operational point can be expressed by a utility function that combines reports from running applications as well as other layers of the protocol stack. For that, cognitive adaptation algorithms include phases such as observation, data analysis, decision-making, and action.

The decisions made by the cognitive engine at the node aim to optimize the protocol stack performance and are driven by the goals specified in the local database. The scope of these goals is local (at node level). Most of them are generated by the demands and the QoS requirements of user applications running at a given cognitive node.

Global optimization goals are defined on an end-to-end basis. The achievement of these goals requires cooperative actions from different network nodes which are implemented using the *cooperation/negotiation plane* operating closely with the cognitive engine to achieve the target goals.

While goals and knowledge databases are directly connected to the cognitive plane of the node and allow instant information exchange, the cognitive plane communication with the protocol stack is performed by the *signaling plane*. The signaling plane is responsible for providing a proper way for the delivery of signaling information delivery. Depending on the signaling type required, for instance, indication of parameter values, signaling threshold violation, or a callback-like indication, different signaling methods are required.

The signaling plane allows information exchange not only between the cognitive engine and different protocol layers of a single node but also provides two interfaces for communication with other network nodes, for enabling the exchange of parameters' values or targeted end-to-end optimization goals. One interface operates on a peer-to-peer basis which allows information exchange between any two nodes of the network in a distributed manner. An alternate (or complementary) one, called *Cognitive Information Service (CIS)*, corresponds to a network broadcast channel where information inserted by a given node is heard by all the nodes of the network segment. CIS signaling has obvious scalability limitations. Because of that, it is mainly used in well-defined parts of the network with limited number of nodes, such as a WiFi cell.

The Cooperation and negotiation plane is responsible for harvesting cognitive information available at other network nodes, filtering and managing them in a distributed manner. Information harvesting can either be scheduled or be pursued by using instant requests or interrupts. Moreover, information could be node-specific or specific to a particular data flow.

The analysis of information gathered from cognitive nodes helps the cognitive engine to construct global knowledge and goals. Upon every adjustment, such information is reported back to cognitive nodes, so that they can adjust their appropriate local databases and, as a result, their behavior.

A main characteristic of the cognitive network architecture is scalability, assured by the use of a combination of centralized (at the node level) and distributed (at the network level) techniques. In particular, at node level, the core cognitive techniques (such as data analysis, decision making, and learning) are concentrated in the cognitive planes of the nodes and implemented in a centralized manner. Furthermore, observation and action software add-ons to the protocol layers serve only as instruments and cognitive planes are typically "non-intelligent". Distributing cognitive process among protocol layers (especially the learning and decision making functions) would require complex algorithms for synchronization and coordination between intra-layer cognitive processes. Alternatively, it seems that a single centralized cognitive process at node level brings a simpler solution, while implementation of cognitive process at network layer must be distributed or clustered implemented.

III. Cooperative Optimization Design

3.1 Inter-Layer Cooperative Optimization

The cooperative optimization framework presented in this section aims at supporting dynamic configuration and optimization of communication protocols. It provides a way for network elements to adapt their configuration and protocol stack parameters in order to constantly adapt the values of protocol parameters to changing network conditions. The process of search for optimal setup of protocol parameters is performed by using cognitive algorithms [10] and by sharing information among network nodes.

The proposed approach is based on the cooperative architecture presented in the previous section and it extensively relies on quality feedback loops as well as on commands allowing the control of internal to the protocol parameters. The core idea

is to enable each node to randomly select minor variations of some parameters, test them and use the information to identify the best parameter setting given the operating context.

The main task of the cognitive plane is the adaptation of different protocol stack parameters in order to converge to an optimal operational point given the network state. This way, the cognitive adaptation algorithms include phases such as data analysis, decision-making, and action, as illustrated in Fig. 2.

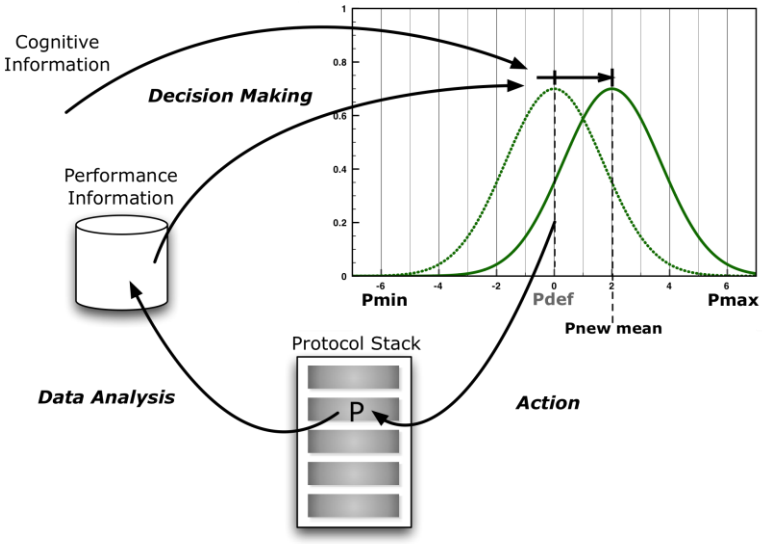

Fig. 2. Cognitive Adaptation Algorithm

One of the main design requirements for the presented cooperative framework is to provide cognitive adaptation with minimal changes in the protocol stack. In the proposed approach, each protocol parameter *P* is expressed in terms of its default value P_{def} and its operation range [P_{min}, P_{max}]. The operation of the protocol is initiated with parameter *P* set to its default value. Then, the cognitive mechanism begins searching for optimal *P* values.

At the end of a given interval *I*, the cognitive mechanism measures using a defined quality metric and stores the obtained performance from the current value of *P* accordingly. Then, the mechanism selects a value of *P* that provides the best performance. That value is assigned to the mean of a normally distributed random number generator. Finally, a new value for *P* is chosen in the range [P_{min}, P_{max}] using the random number generator. The initial mean for the number generator is P_{def}. This loop continuously adjusts the mean of the normal distribution to the value of *P* that provides the best performance under the current network scenario. The mean of the normal distribution converges to the best *P* value for the current network state. The standard deviation assigned to the normal distribution affects the aggressiveness of the mechanism in trying new values of *P* at each interval *I*.

3.2 Inter-Node Cooperative Optimization

This section extends the previous section by adding the dimension of inter-node cooperation which allows network nodes to exchange available cognitive information as well as to provide the means for making joint decisions.

Similar to the signaling plane, the Cognitive Information Service (CIS) provides the means for cognitive information exchange and its aggregation among network nodes.

Fig. 3 shows the CIS implementation in a corporate network segment. CIS servers may become a bottleneck if overloaded since they provide centralized solutions for cognitive information management and aggregation. Consequently, they should be used in well-defined network segments with limited number of nodes.

Taking into account that the cognitive communications in the direction from the CIS server to the cognitive nodes are often point-to-mulitipoint, they can be implemented either at the IP level using broadcast protocols or at the link layer. The layer-2 implementation will bring efficiency in cognitive information exchange and lead to the reduction of signaling overhead.

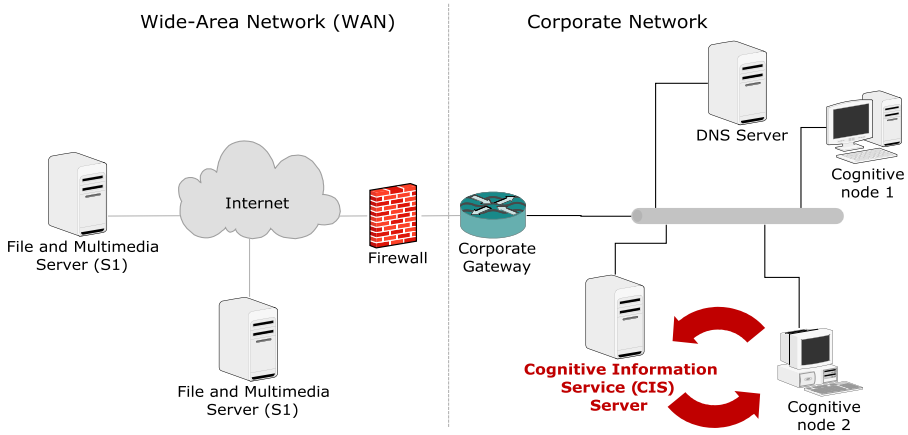

Fig. 3. Cognitive Information Service in cooperative network segment.

The configuration offered by CIS to a new node will not necessarily be optimal since the ideal configuration was inferred without information on running applications, traffic demands, and the peers the node is willing to communicate with. However, such configuration can potentially offer better startup performance than using fixed default values for the protocols, as is currently done for the protocols of the TCP/IP stack.

Depending on the nature and requirements for information exchange, three different signaling methods can be used: in-band signaling, on-demand signaling, and broadcast signaling.

In-band signaling is the most effective signaling method from the point of view of overhead reduction. Cognitive information can be encapsulated into ongoing traffic flows, for example into optional packet header fields [9], and delivered without waste of bandwidth resources.

Due to low overhead, in-band implementation of CIS is best suited for networks with wireless technologies used for access networks such as WiFi, WiMAX, and cellular, where the bottleneck of the end-to-end connection is typically at the wireless link.

Another advantage of in-band signaling is that cognitive information can be associated with the portion of application data it is delivered with.

However, the main shortcoming of in-band signaling is the limitation of signaling to the direction of the packet flow, making it not suitable for cognitive schemes requiring instant communication between nodes having no ongoing data exchange.

On-demand signaling method operates on a request-response basis and can be complementary to in-band signaling. It is designed for cases requiring instant cognitive information delivery between network nodes. Cognitive information becomes available at the requesting node following round-trip time delay, which makes it well suited for Wired/Wireless LAN scenario.

One of the core signaling protocols considered in on-demand signaling is the Internet Control Message Protocol (ICMP). Generation of ICMP messages is not constrained by a specific protocol layer and can be performed at any layer of the protocol stack. However, signaling with ICMP messages involves operation with heavy protocol headers (IP and ICMP), checksum calculation, and other procedures which increase processing overhead.

Broadcast signaling method allows point-to-multipoint cognitive information delivery from CIS server to the network nodes located in the same segment, while keeping low overhead. Broadcasting is especially suited for wireless networks following cellular organization.

Cognitive information is encapsulated into a beacon periodically broadcasted by wireless gateways (access points or base stations), and thus fits scenarios where cognitive information delivery is tolerant to delays and can be performed at regular intervals.

IV. A TEST CASE: TCP OPTIMIZATION USING A COOPERATIVE FRAMEWORK

4.1 IMPLEMENTATION

In order to present the benefits of the proposed cooperative optimization framework we extended the Network Simulator (ns2) [12] with the required functionalities.

Simulated network topology is presented in Fig. 4. It consists of four cognitive nodes *S* running intra-layer cognitive engine and CIS server performing inter-node cognitive operations all connected using 100 Mb/s, 0.1 ms links in a star topology centered at router R_1. Such connectivity aims at mimicking operation of Ethernet network segment. Similar configuration is followed by the destination nodes *D*, which do not implement any cognitive functionality.

Routers R1 and R2 are connected with four links with propagation delays of 10ms, 50ms, 100ms, and 200 ms, and Packet Error Rates (PERs) of 0.0, 0.05, 0.1, and 0.15.

A maximum of four flows can be started in the simulated topology between nodes corresponding S and D. For example, the first flow is initiated between S_1 and D_1 and flows through L_1 link.

Different values of link PERs will require different window increase strategies, controlled by the parameter α, so that optimal performance can be achieved. High values of α are expected to bring better throughput for high PERs. However, in case

of no errors, high values of α will lead to multiple congestion-related losses and throughput degradation due to retransmission.

Different RTTs are designed to influence the choice of both α and β parameters through different time required for S nodes to react to congestion- or link-related losses.

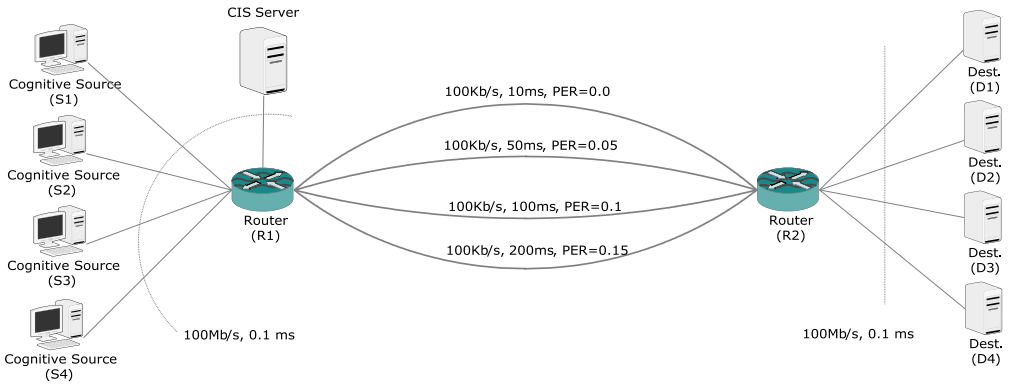

Fig. 4. Simulated topology.

The Network Simulator (ns2) module that implemented the Cooperative framework focused on the cognitive adaptation of the main parameters controlling TCP protocol steady state behavior, i.e. speed of window increase α and aggressiveness of multiplicative decrease β. The main performance feedback metric is the end-to-end TCP goodput which is the amount of data successfully delivered to the receiver.

At the beginning of each flow, α and β are set to their default values 1 and 0.5, respectively. Then, right after the end of the TCP slow start phase, a timer is started for guiding the cognitive engine implemented inside each network node which corresponds to intra-layer cognitive functionality. Whenever the timer expires, intra-layer cognitive engine performs the steps defined by Algorithm 1. First, it computes, analyses, and stores the throughput value. Then, it selects α and β values corresponding that lead to optimal throughput values and initializes α and β with the mean values given by the normally distributed random number. Finally, samples are taken from these generators are used to obtain α and β values that will be used in the next sampling interval.

The measured performance in terms of TCP goodput is averaged using an exponentially weighted moving average as follows:

$$T_a = T_a * (w) + T_m * (1 - w),$$

where *w* is the weight assigned to the average goodput (*Ta*) computed for the corresponding value of α. Our experiments showed that *w* = 0.5 provides a good tradeoff between current and past values of the goodput.

The inter-node level implementation of the cognitive engine is presented in Algorithm 2 and it follows a similar approach. At regular intervals, each cognitive node communicates the chosen α and β values and the throughput in the immediate past interval. Communication is pursued using the ICMP protocol. Moreover, typically CIS service is implemented in the same network segment of cognitive nodes and should not lead to performance degradation of the data flows. Nevertheless, the

interval used for cognitive nodes to report to CIS should be chosen carefully considering possible overhead issues.

Algorithm 1: Intra-layer Cognitive Engine

Analyze Performance Metrics

1. **Get the** number of bytes received since last time interrupt nBytes
2. **Get** the time elapsed from the last timer interrupt smapleInterval
3. **Calculate the** average throughput R_i as nBytes/sampleInterval
4. **Calculate the** weighted throughput $R_i = R_{i-1} \times (p) + R_i \times (1-p)$, where p is the weight value given to the past history
5. **Store the** weighted Ri value in two-dimensional array on a position defined by the current α and β values [α, β]

Get parameters corresponding to optimal performance

6. **Set** Max Throughput to zero
7. **For** each element of the two-dimensional array with Ri **do**
8. **If** current throughput Ri is greater than Max Throughput **then**
9. **Set** Max Alpha equal to α
10. **Set** Max Beta equal to β
11. **Endif**
12. **Endfor**

Choose parameters for next sampling interval

13. **Set** Normal Distribution Mean to Max Alpha
14. **Get** new α value from the random number
15. **Set** Normal Distribution Mean to Max Beta
16. **Get** new β from the distribution

Algorithm 2: Inter-node Cognitive Engine

Receive feedback and obtain cumulative throughput maximum

17. **Set** the Total Throughput R_{TOT} equal to zero
18. **For** each cognitive node **do**
19. Receive α, β values, and the measured Throughput Ri
20. Increase R_{TOT} to Ri
21. **Endfor**
22. **Store** R_{TOT} along with α and β parameters to each node
23. **Get** max R_{TOT} and α_{max} and β_{max} parameters corresponding for every cognitive node involved into R_{TOT} calculation

Configure each cognitive node parameters

24. **For** each cognitive node **do**
25. **If** current α is not equal to α_{max} **then**
26. **Set** α equal to α_{max}
27. **Endif**
28. **If** current β is not equal to β_{max} then
29. Set β equal to β_{max}
30. **Endif**
31. **Endfor**

4.2 INTER-LAYER COGNITIVE OPTIMIZATION

In order to obtain the details of inter-layer cognitive optimization, we first limited our scenario to only two TCP connections simulated in separate experiments: flow F_1 between S_1 and D_1 involving L_1 link and flow F_3 between S_3 and D_3 involving L_3. The average flow throughput is chosen as the main performance metric, which is measured for different values of α with β fixed at 0.5. We used 1s as sampling interval for the cognitive engine and 0.5 for standard deviation of normal distributed random numbers.

Results presented in Fig. 5 show the ability of the cognitive engine to converge to the optimal value of parameter α. For F_1, the optimal value of α is equal to 1 due to the absence of link errors - since every packet loss is caused by congestion; while for F_3 the optimal value of α is 4, which corresponds to a balance between increase rate after loss related to link error and the amount of retransmissions performed due to multiple congestion-related losses at the bottleneck buffer.

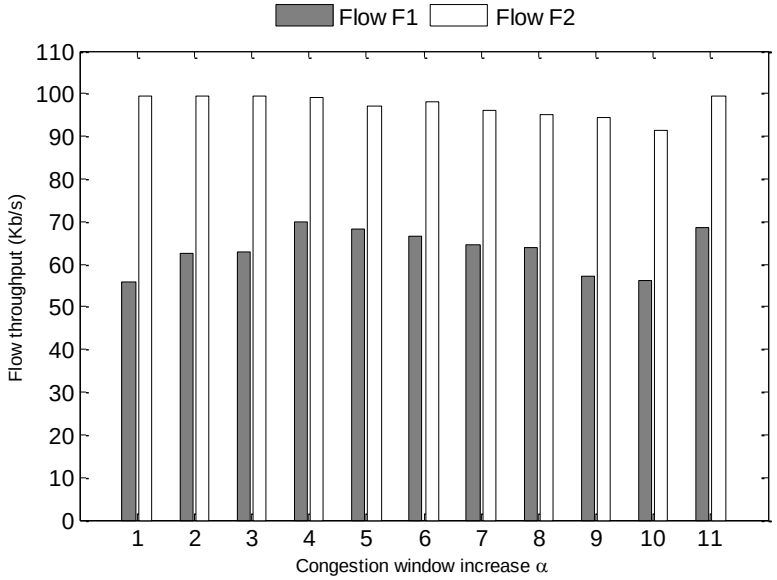

Fig. 5. Average throughput performance of a single TCP flow with different fixed congestion window increase (α) parameters and cognitive adaptation approach.

Fig. 6 presents the distribution of the α values chosen by intra-layer cognitive engine during a TCP flow lifetime. Each value is obtained by dividing the time at which TCP flow congestion control used a given value of alpha divided by the total simulation time. As expected for the flow F_1 most of the chosen α values are gathered around α=1, while for flow F_3 α=4, most of the time.

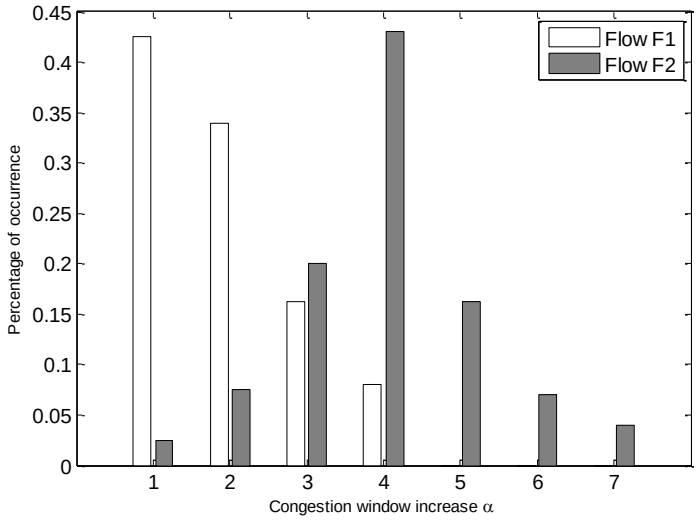

Fig. 6. Density of congestion window increase (α) parameters chosen by intra-layer cognitive engine.

Results confirm that the proposed cognitive adaptation leads to significant improvements in a dynamic network environment by performing both intra-layer and inter-layer optimizations. It was shown that fixing the α value performance degradation under specific scenarios can happen. The proposed cognitive mechanism avoids that problem. If there were no global optimal values for a protocol parameter, certainly the presented cognitive adaptation can provide the best average performance by adapting the protocol behavior to current network conditions.

4.3 INTER-NODE COGNITIVE OPTIMIZATION

Previous section showed the benefits of using intra-layer cognitive engine for tuning the performance of a single TCP flow. In this section, we focus on joint optimization of the performance of multiple TCP flows. To this aim, we simulated four TCP NewReno flows between specific S and D pair of nodes as illustrated in Fig. 4.

Fig. 7 presents the throughput results for each individual flow as well as the cumulative throughput for the following three cases: i) no cognitive adaptation, ii) with intra-layer cognitive adaptation, and iii) with inter-node cognitive adaptation (CIS server). In both cases when cognitive adaptation is used, α and β TCP flow control parameters are tuned as outlined in the section above.

As expected, the throughput decreases for long links with high error rates. The main reasons for such throughput reductions are link errors and well-known RTT unfairness for flows with different RTTs competing for the same buffer resources.

However, it can be observed that intra-layer cognitive engine can easily solve the problem of link losses by adapting α and β parameter, converging to the optimal throughput value. However, it cannot cope with the problem of RTT unfairness, which requires coordination between flows. Such coordination is performed at the inter-node level by the CIS server, which leads to higher performance.

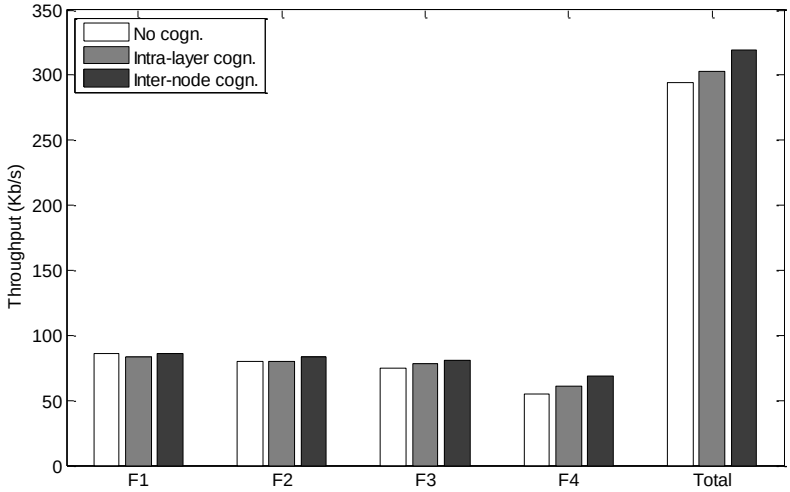

Fig. 7. Multiflow TCP throughput performance for case with no cognitive adaptation, intra-layer cognitive adaptation, and inter-node cognitive adaptation.

V. CONCLUSIONS

This chapter introduces a novel concept of cooperative network optimization that is carried at inter-layer and inter-node basis. Based on the proposed concept, protocols from the TCP/IP can be extended to dynamically tune their configuration parameter values based on the past performance.

Results demonstrated that cooperation between the protocol layers of a protocol stack can significant enhance the performance when compared to a fixed non-cooperative approach. Moreover, the exchange of information on configuration and performance among network nodes can further improve the performance of data transfer.

One of the main points behind the design of the cooperative optimization framework is the ability to tune configuration parameters in runtime and in a distributed manner. This ensures fast convergence and optimal protocol stack performance adaptation in dynamically changing network environments.

REFERENCES

[1] J. Mitola III, “Cognitive radio for flexible mobile multimedia communications,” Mobile Networks and Applications, vol. 6, no. 5, September 2001.

[2] M. W. Murhammer and E. Murphy, “TCP/IP: Tutorial and Technical. Overview,” Upper Saddle River, NJ: Prentice-Hall, 1998.

[3] J. Postel and J. Reynolds, “File Transfer Protocol (FTP),” RFC 959, IETF, October 1985.

[4] Clark, George C., Jr., and J. Bibb Cain, “Error-Correction Coding for Digital Communications,” New York: Plenum Press, 1981, ISBN 0-306-40615-2.

[5] ITU-T, P.800, “Methods for Subjective Determination of Transmission Quality,” Aug. 1996.

[6] J. Postel, “Transmission Control Protocol,” RFC 783, September 1981.

[7] J. Postel, “User Datagram Protocol,” RFC 768, Aug. 1980.

[8] IEEE 802.11 Wireless Local Area Networks. Available from: http://grouper.ieee.org/groups/802/11/

[9] ANWIEEE Std 802.3, “Carrier Sense Multiple Access with Collision Detection,” 1985.

[10] K. Machova and J. Paralic, “Basic Principles of Cognitive Algorithms Design. Proceedings of the ICCC International Conference Computational Cybernetics,” Siofok, Hungary, 2003.

[11] T. Kelly, “Scalable TCP Improving performance in highspeed wide area networks”, Computer Communication Review vol. 32, no. 2, April 2003.

[12] The network simulator ns2. Available from: http://www.isi.edu/nsnam/ns.